Git Stash Pop: A Comprehensive Guide

The git stash pop command is used to **reapply stashed changes** to your working directory and **remove the stash from the stash list**. It is a combination of git stash apply (which reapplies stashed changes without removing them) and git stash drop (which deletes the stash entry).

Why Use git stash pop?

1. Temporarily save work-in-progress changes when switching branches or working on a new task.
2. Reapply those saved changes later and remove them from the stash.

How git stash pop Works

1. **Save Work-in-Progress (WIP):**
    - You make changes to your working directory but don't want to commit yet.
    - Use git stash to temporarily save these changes.
2. **Switch Context:**
    - You switch branches or tasks to address something else.
3. **Reapply Stashed Changes:**
    - After completing the other work, you return and use git stash pop to reapply the changes and remove the stash entry.

Basic Workflow with git stash pop

1. **Stash Changes:**

git stash

This saves your current uncommitted changes (both staged and unstaged) and restores the working directory to the last commit.

2. **Check Stash List:**

git stash list

This shows all saved stashes.

3. **Apply and Remove the Most Recent Stash:**

git stash pop

Examples

Example 1: Stashing Changes While Switching Branches

Step 1: Make Changes in Your Branch

```
echo "Temporary work" > temp.txt
```

```
git add temp.txt
```

Step 2: Save Changes to the Stash

```
git stash
```

- Your changes are now saved, and the working directory is clean.

Step 3: Switch to Another Branch

```
git checkout main
```

Step 4: Perform Work in main and Commit

```
echo "Bug fix on main" > fix.txt
git add fix.txt
git commit -m "Bug fix on main branch"
```

Step 5: Switch Back to the Original Branch

```
git checkout feature
```

Step 6: Reapply Stashed Changes and Remove Them

```
git stash pop
```

- Your previous work (temp.txt) is restored, and the stash is removed from the list.

Example 2: Conflict Resolution During git stash pop

If the stashed changes conflict with the current working directory, Git will pause the stash pop process and report conflicts.

Scenario:

1. File example.txt exists in both the stash and the current branch, but with different changes.
2. When running git stash pop, a conflict occurs.

Steps:

1. **Create a Conflict**:
    - Modify example.txt and stash it:

```
echo "Work in progress" >> example.txt
git add example.txt
git stash
```

- Modify example.txt again in the current branch and commit it:

```
echo "Different changes" >> example.txt
git add example.txt
git commit -m "New changes in current branch"
```

2. **Reapply Stash and Encounter Conflict**:

```
git stash pop
```

Git reports a conflict in example.txt:

Auto-merging example.txt

CONFLICT (content): Merge conflict in example.txt

3. **Resolve the Conflict**:
   - Open example.txt, manually resolve the conflict, and save the file.
   - Stage the resolved file:

```
git add example.txt
```

4. **Complete the Stash Pop**:

```
git stash drop
```

If you didn't manually remove the stash earlier, Git automatically drops the stash after a successful pop.

Example 3: Applying a Specific Stash

If you have multiple stashes, you can use git stash pop with a specific stash identifier.

Step 1: Create Multiple Stashes

```
echo "Change 1" > file1.txt
git add file1.txt
git stash


echo "Change 2" > file2.txt
git add file2.txt
git stash
```

Step 2: Check Stash List

```
git stash list
```

Output:

```
stash@{0}: WIP on feature: Change 2
stash@{1}: WIP on feature: Change 1
```

Step 3: Pop a Specific Stash

```
git stash pop stash@{1}
```

- The changes from stash@{1} are reapplied, and that stash is removed.

Key Commands

Command	Description
git stash	Save current changes to the stash.
git stash list	List all stashed changes.
git stash pop	Apply the most recent stash and remove it from the stash list.
git stash pop stash@{n}	Apply and remove a specific stash identified by stash@{n}.
git stash apply	Apply the most recent stash without removing it from the stash list.
git stash drop	Remove a specific stash from the stash list.
git stash clear	Remove all stashes from the stash list.

When to Use git stash pop

1. **Quick Work Context Switch**: Use it to temporarily save incomplete work when switching to another task.
2. **Restoring WIP Safely**: Use it to restore changes once you've returned to the original branch.
3. **Single-Use Stash**: Use git stash pop when you don't need to keep the stash after applying it.

Tips and Best Practices

1. **Use git stash list Regularly**: Keep track of stashes to avoid confusion when working with multiple stashes.
2. **Use Descriptive Messages**: Add messages to your stashes for clarity:

git stash save "Added initial code for feature X"

3. **Be Cautious with Conflicts**: Always resolve conflicts carefully to avoid losing work.
4. **Avoid Stashing Too Much**: Don't rely on stashing for long-term storage. Commit changes frequently.